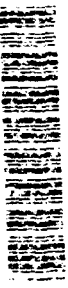


AD-A258 253



Special Report

CMU/SEI-92-SR-10



Carnegie-Mellon University
Software Engineering Institute

2

**A Bibliography of Externally Published Works
by the SEI Engineering Techniques Program**

**Sandy Brenner
Gibbie Hart**

August 1992

**DTIC
ELECTE
DEC 10 1992
S E D**

~~RESTRICTED~~ STATEMENT
**Approved for public release;
Distribution Unlimited**

125430
92-31231



UBPS

92 12 09 058

Special Report
CMU/SEI-92-SR-10
August 1992

A Bibliography of Externally Published Works by the SEI Engineering Techniques Program



Sandy Brenner

Engineering Techniques Program

Gibbie Hart

Products and Services Planning

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

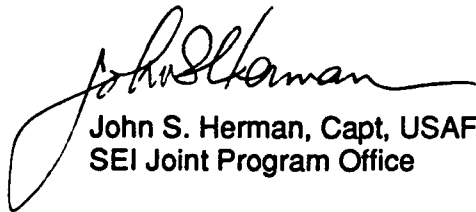
SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER



John S. Herman, Capt, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Copyright © 1992 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

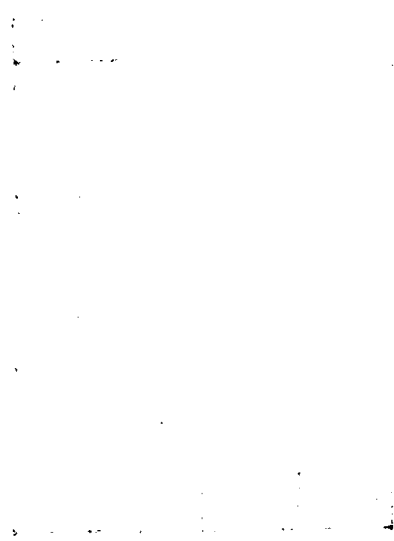
Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

1	Introduction	1
2	Journal and Magazine Articles	3
3	Books in Part or in Whole	8
4	Work in Proceedings	12
5	Acknowledgments	27
Appendix A Information about the Software Engineering Institute		29
Appendix B Members of the SEI Engineering Techniques Program as of July 1992		31
Index by Author		33
Index by Target Audience		35

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

DTIC QUALITY INSPECTED 2



A Bibliography of Externally Published Works by the SEI Engineering Techniques Program

Abstract: This bibliography lists works by the members of the Software Engineering Institute (SEI) Engineering Techniques Program that are not published or available from the SEI. The bibliography is organized by type of work (i.e., journal or magazine article, book, or proceedings). Indices are provided by author and targeted audience.

1 Introduction

This bibliography lists work by members of the Engineering Techniques Program that were published during their employment at the SEI, but are not available from the SEI.

The organization of the bibliography is by type of work (journal or magazine article, books in whole or in part, and work in proceedings). Within each of these categories, the organization is alphabetical by primary author. Each entry contains the name(s) of the author(s), title of the work, targeted audience, and publication information. Where available, abstracts are also included.

The primary focus of the Engineering Techniques Program is tools and methods that operate on and within the software process to control and improve it. Some of the subjects that are addressed in the works listed are:

- CASE tools, adoption, and integration
- code inspection
- computing research and direction
- configuration management
- domain analysis
- domain-specific architectures
- integrated software process environments (IPSEs)
- interactive video, multi-media, and virtual reality
- model-based engineering
- programming languages
- reuse
- software architectures
- software engineering as a discipline
- software engineering education
- software engineering environment integration

The target audiences were determined using the following definitions:

Manager: Senior managers responsible for leading software organizations to build higher quality software and to do it more productively.

Practitioner: Software professionals responsible for producing and maintaining better software.

Educator: Those responsible for improving software engineering education and training.

More information about the Software Engineering Institute is provided in Appendix A. Appendix B provides a listing of the members of the Engineering Techniques Program as of July 1992.

Note: To obtain copies of the publications listed in this document, contact your local libraries or the sources cited.

2 Journal and Magazine Articles

- [Barbacci85] Barbacci, M.; Habermann, A.N.; Shaw, M. "The Software Engineering Institute: Bridging Practice and Potential." *IEEE Software* 2, 6 (November 1985): 4-21.
Audience: Manager
Abstract: The state of the practice in software engineering has not evolved fast enough to keep pace with the rising demand for increasingly larger, more sophisticated software systems and components. Today, software contractors are faced with numerous difficulties that prevent them from fulfilling the software needs of the market—difficulties that result in excessive costs for systems with reliability that is, in too many cases, questionable at best.
- [Brown et al 92b] Brown, A.W.; McDermid, J.A. "Learning from IPSEs Mistakes." *IEEE Software* 9, 2 (March 1992): 23-28.
Audience: Practitioner
Abstract: A discussion of some of the issues of tool integration in an integrated project support environment (IPSE). A conceptual framework is proposed for analyzing and measuring aspects of integration in an IPSE.
- [Brown et al 92d] Brown, A.W.; Penedo, M.H. "An Annotated Bibliography on Software Engineering Environment Integration." *ACM Software Engineering Notes* 17, 3 (July 1992): 47-55.
Audience: Manager, Practitioner
Abstract: A detailed bibliography on tool integration in a software engineering environment. Contains summaries of over 20 pages that are important in to this topic area.
- [Christel92a] Christel, M.G. "Virtual Reality on a PC." *Instruction Delivery Systems* 6, 3 (July/August 1992): 6-9.
Audience: Practitioner, Educator
Abstract: A digital video course on software technical reviews illustrates how virtual reality can be implemented today on a personal computer platform. The advantages of using a virtual environment for teaching code inspections are discussed, as are the implications of making that environment increasingly realistic.
- [Dart89e] Dart, S.A. "Tool Configuration Assistant." *ACM SIGSOFT Software Engineering Notes* 17, 7 (November 1989): 110-113.
Audience: Practitioner

- [Huff92] Huff, C.C. "Elements of a Realistic CASE Tool Adoption Budget." *Communications of the ACM* 35, 4 (April 1992): 45-54.
Audience: Manager, Practitioner
Abstract: All too often would-be CASE implementors find that the costs of adopting a new CASE tool are much higher than they originally anticipated. This is because implementors often focus only on the direct CASE tool purchase cost, overlooking significant other costs – not only in dollars, but in terms of people and time. This article attempts to address the very practical issue of providing a complete basis for a realistic cost estimate for the adoption of a CASE. A summary of this article is organized into two tables: primary cost items and primary cost drivers. Many subjects discussed in this article lead to other significant CASE-related topics. These are addressed superficially, but pointers are supplied to more detailed information.
- [Kaiser88a] Kaiser, G.E.; Feiler, P.H. "Intelligent Assistance for Software Development and Maintenance." *IEEE Software* 5, 3 (May 1988): 40-49.
Audience: Practitioner, Educator
- [Kaiser88b] Kaiser, G.E.; Barghouti, N.S.; Feiler, P.H.; Schwanke, R.W. "Database Support for Knowledge-Based Engineering Environments." *IEEE Expert* 3, 2 (May 1988): 18-23, 26-32.
Audience: Practitioner
- [Klein89b] Klein, D.V. "Comparison of RISC CPUs." *MIPS – The Magazine of Intelligent Personal Systems* 1, 7 (July 1989): 52-56.
Audience: Practitioner
- [Peterson91] Peterson, A.S. "Coming to Terms with Software Reuse Terminology: A Model-Based Approach." *ACM SIGSOFT Software Engineering Notes* 16, 2 (April 1991): 45-51.
Audience: Practitioner, Educator
Abstract: This article attempts to standardize the use of many terms used in the software reuse literature. Three terms of particular interest – taxonomy, software reuse, and domain analysis – and some problems with their usage are discussed. The specific problems with these terms are generalized and several solutions are given, the most important being the introduction of the concept of using reuse process models to provide both context and an overall view of the potential areas of discourse in reuse. Several new terms are proposed for future use; definitions of existing terms that are meaningful in the context of software reuse are also included.

- [Schaefer92] Schaefer, W.; Shaw, M. "Design Methods and Software Processes." *ACM Press Software Engineering Notes* 17, 1 (January 1992): 44-51.
Audience: Practitioner, Educator
- [Shaw86b] Shaw, M. "Beyond Programming-in-the-Large: The Next Challenges for Software Engineering." *IFIP Newsletter* 3, 2 (June 1986): 1, 5.
Audience: Manager, Practitioner, Educator
Abstract: As society's dependence on computing broadens, software engineering is being called upon to address new problems that raise new technical and nontechnical concerns. Aspirations and expectations for the application of computers appear to be unbounded, but present software development and support techniques will not be adequate to build computational systems that satisfy our expectations, even at very high cost. Each order-of-magnitude increase in the scale of the problems being solved leads to a new set of critical problems that require essentially new solutions. The next challenges for software engineering will deal with software as one of many elements in complex systems, which we call program-as-component, and with the role of software as an active participant in the software development process, which we call program-as-deputy.
- [Shaw87e] Shaw, M. "Poza Programowanie Wielkoskalowe - Kolejne Wyzwania Dla Inzynierii Oprogramowania" (Polish translation of "Beyond Programming-in-the-Large: The Next Challenges for Software Engineering"). *Informatyka* 22, 7 and 8 (July and August 1987): 1-4 and 8-11.
Audience: Manager, Practitioner, Educator
Abstract: See Shaw86b, above.
- [Shaw88a] Shaw, M. "The Impact of Abstraction Concerns on Modern Programming Languages." Published by Iwanami Shoten, Publishers of Tokyo, Japan. Japanese translation and reprint from *IEEE Software* 5, 7 (August 1988): 47-62.
Audience: Practitioner, Educator
Abstract: See Shaw87b, page 9.
- [Shaw89b] Shaw, M. "Remembrances of a Graduate Student." *Annals of the History of Computing, Anecdotes Department* 11, 2 (1989): 141-143.
Audience: Educator
Abstract: Just as some years are memorable for their wines, some years are memorable for their research. In software, 1968 was such a year. It was the year the software research community really started thinking systematically, even formally, about software structures.
- [Shaw89e] Shaw, M. "Larger Scale Systems Require Higher-Level Abstractions." (Revised version of "Toward Higher-Level Abstractions for Software Systems") *Software Engineering Notes* 14, 3 (May 1989): 143-146.
Audience: Practitioner, Educator
Abstract: See Shaw89d, page 22.

- [Shaw90d] Shaw, M. "Toward Higher-Level Abstractions for Software Systems." *Data and Knowledge Engineering* 5, 2 (1990): 119-128.
Audience: Practitioner, Educator
Abstract: See Shaw86b, page 5.
- [Shaw90f] Shaw, M. "Seeking a Foundation for Software Engineering." Interview. *IEEE Software* 7, 2 (March 1990): 102-103.
Audience: Manager, Practitioner, Educator
Abstract: Can history provide at least a rough guide to the nature of the problems that face software engineering? An engineering discipline emerges from craftsmanship in two stages, Shaw has found. First, systematic production techniques and management methods turn craftsmanship into routine commercial production, as shown in the accompanying table. Later, the problems of production practice yield professional engineering practice. In an interview with Contributing Editor Ware Myers, Shaw described this process as it relates to software.
- [Shaw90h] Shaw, M. "Prospects for an Engineering Discipline." *IEEE Software* 7, 6 (November 1990): 15-24.
Audience: Manager, Practitioner, Educator
Abstract: Software engineering is not yet a true engineering discipline, but it has the potential to become one. Older engineering fields offer glimpses of the character software engineering might have. The term "software engineering" was coined in 1968 as a statement of aspiration—a sort of rallying cry. That year NATO convened a workshop by that name to assess the state and prospects of software production. The resulting practice, however, differs significantly from the practice of older forms of engineering.
- This paper examines the usual practice of engineering and the way it has evolved in other disciplines, considering two examples in depth. This provides a historical context for assessing the current practice of software production and setting out an agenda for attaining an engineering practice.
- [Shaw91a] Shaw, M. "Informatics for a New Century: Computing Education for the 1990s and Beyond." *Education and Computing* 7, 1-2 (1991): 9-17.
Audience: Educator
Abstract: Information technology and computer science have not only reshaped computation, communication, and commerce; they have expanded the basic models and paradigms of many disciplines. Informatics education has obligations to all the communities that rely on information technology, not just computing professionals. Serving this extended audience well requires changes in the content and presentation of computing curricula. This paper sketches the coming needs for information processing and analyzes the populations that will require

informatics education. It considers curriculum requirements through two examples, one outside the traditional boundary of computer science and one inside.

[Shaw92a]

Shaw, M. "We Can Improve the Way We Teach CS Students." Letter to the Editor. *Computing Research News* 4, 1 (January 1992): 2-3.

Audience: Educator

Abstract: Hurrah for William A. Wulf's statement in opposition to separate undergraduate software engineering programs.

[Smith90]

Smith, D.; Oman, P. "Software Tools in Context." *IEEE Software* 7, 5 (May 1990): 14-20.

Audience: Manager, Practitioner

Abstract: This article discusses the background for environments and tools. It overviews the major types of tools and the place of tools within software development. It discusses some of the issues in the use of tools, such as integration and scalability, and the implications of tools for software development.

[Stevens92b]

Stevens, S.M. "Time Traveler." *Pixel: The Magazine of Visualization* 3, 3 (March/April 1992): 34-39.

Audience: Practitioner

Abstract: This article discusses holographic and optical techniques to create real (in the optical sense) three dimensional interactive images. Interactive design criteria and human factors based visual fidelity issues are addressed.

[Stevens89]

Stevens, S.M. "Intelligent Interactive Video Simulation of a Code Inspection." *Communications of the ACM* 32, 7 (July 1989): 832-843. (Translated into Japanese and published in *Nikkei Artificial Intelligence* 1, 20H, 1990.)

Audience: Practitioner, Educator

Abstract: The need for technological solutions to learning in the software engineering field is increasing. The Advanced Learning Technologies Project (ALT) has developed a highly interactive, high-fidelity simulation of group process communication. The first course demonstrating these techniques is on the formal technical review known as code inspection. This article discusses technical aspects of ALT's system design.

3 Books in Part or in Whole

[Brookes et al 85]

Brookes, S.D.; Donner, M.; Driscoll, J.; Mauldin, M.; Pausch, R.; Scherlis, W.L.; Shaw, M.; Spector, A. "Final Report of Curriculum Design Group: Stephen D. Brookes, Marc Donner, James Driscoll, Michael Mauldin, Randy Pausch, William L. Scherlis, Mary Shaw, and Alfred Spector," Mary Shaw, editor. *The Carnegie-Mellon Curriculum for Undergraduate Computer Science*. New York: Springer-Verlag, 1985.

Audience: Educator

Abstract: Reflecting the structure and emerging needs of modern computer science, this book presents a comprehensive redesign of the traditional curriculum emphasizing: the integration of fundamental conceptual material with the best of current practical applications; the identification and reorganization of topics based on common themes into single new courses; and the specification of requirements for computer support needed to illustrate important points made in the courses described.

The Carnegie-Mellon curriculum design, developed by eight leading computer scientists, is formulated to last for the next decade and to play a leadership role in computer science education. Specifically, the curriculum includes: a detailed descriptions of 30 new or revised computer science courses; requirements for a computer science major consisting of a small core of courses that allow for a variety of specializations within the major; courses offered by other departments that present material relevant to computer science.

[Brown et al 92a]

Brown, A.W.; Earl, A.N.; McDermid, J.A. *Software Engineering Environments: Automated Support for Software Engineering*. Maidenhead, England: McGraw-Hill, 1992.

Audience: Practitioner, Educator

Abstract: Over 350 pages describing requirements for a software engineering environment (SEE), an analysis of a number of existing SEEs, and an examination of some key issues in this field. The book makes use of the European Computer Manufacturers (ECMA) reference model for frameworks of a SEE as a basis for comparing HP SoftBench, ECMA PCTE, DEC's CIS, and IBM's AD/Cycle.

[Cohen90a]

Cohen, S.G.; Palmer, C. *Engineering and Application of Reusable Software Resources*. Washington, DC: American Institute of Aeronautics and Astronautics, 1990.

Audience: Manager, Practitioner

Abstract: The potential benefits of software reuse are well-known; thus the question arises "Why isn't software reuse more widely practiced?" A misconception about what is required to encourage effective levels of reuse has hindered reuse efforts in the past. It is not reason-

able to expect high levels of reuse to result from merely cataloging existing code and making it available. Software must be engineered for reuse (either initially or during some type of retry-fitting process), and new applications must plan for reuse. There are three approaches to software reuse that cover the spectrum from short-term to long-term: adaptive, parameterized, and engineered. Tool support can facilitate both the introduction and practice of software reuse.

[Dart91a]

Dart, S.A.; Druffel, L. "Software Development Environments," 203-227 *American Institute of Aeronautics and Aerospace (AIAA), Progress Series on Aerospace Software Engineering*, Washington, DC: AIAA, 1991.

Audience: Manager, Practitioner, Educator

[Shaw87a]

Shaw, M. "Education for the Future of Software Engineering," 344-357. *Software Engineering Education: The Educational Needs of the Software Community*. New York: Springer-Verlag, 1987.

Audience: Educator

Abstract: The discipline of software engineering is developing rapidly. Its practitioners must deal with an evolving collection of problems and with new technologies for dealing with those problems. Software engineering education must anticipate new problems and technologies, providing education in the enduring principles of the field in the context of the best current practice. Since changes in the discipline cannot be completely anticipated, software engineers must be able to assume responsibility for their own continuing professional development. This paper describes significant changes now taking place in the field of software engineering and proposes some goals and objectives for the professional education of software engineers.

[Shaw87b]

Shaw, M. "The Impact of Abstraction Concerns on Modern Programming Languages." Reprinted in IEEE Tutorial. Edited by Gerald E. Peterson. *Object-Oriented Computing, Volume II: Implementations*. Silver Spring, MD: Computer Society Press of the IEEE, 1987.

Audience: Practitioner, Educator

Abstract: The major issues of modern software are its size and complexity, and its major problems involve finding effective techniques and tools for organization and maintenance. This paper traces the important ideas of modern programming languages to their roots in the problems and languages of the past decade and shows how these modern languages respond to contemporary problems in software development. Modern programming's key concept for controlling complexity is *abstraction*—that is, selective emphasis on detail; new developments in programming languages provide ways to support and exploit abstraction techniques.

[Shaw87c]

Shaw, M. "The Impact of Abstraction Concerns on Modern Programming Languages." (*before revision titled: "The Impact of Abstraction Concerns on Modern Programming Languages"*), 232-247. Edited by Marvin V. Zelkowitz. Reprinted in *Selected Reprints in Software, 3rd edition*, Washington, DC: IEEE Computer Society Press, 1987.

Audience: Practitioner, Educator

Abstract: See Shaw87b, page 9

[Shaw89a]

Shaw, M. "Maybe Your Next Programming Language Shouldn't Be a Programming Language," 75-82. *Scaling Up: A Research Agenda for Software Engineering*. Washington, DC: National Academy Press, 1989.

Audience: Practitioner, Educator

Abstract: Software needs now strain the design limits of traditional programming languages. Modern application needs are not satisfied by traditional programming languages, which evolved in response to systems programming needs. Current programming language research focuses on incremental improvements, not on major changes to the nature of software development. But major breakthroughs are needed in two areas:

1. Non-programmers dominate modern computer use. Low computing costs have enabled a wide spectrum of application, and end users who are not programmers need to control their own computations. Order-of-magnitude increases in service require substantial shifts of technology. Computer users are interested in results, not in programming; software must reflect this.

2. Requirements for large complex software systems exceed our production ability. Growth of demand is greater than growth in capacity, and system requirements exceed the scope of conventional languages. Software lacks a true engineering base. Language concepts can support a design level above the algorithm/data structure level and contribute to an engineering discipline.

Programming language designers must look beyond the traditional systems programming domain and tackle problems of special-purpose software and system-level software design.

[Shaw90a]

Shaw, M. "The Impact of Abstraction Concerns on Modern Programming Languages," 139-155. Reprinted in IEEE Reprint Collection. Edited by Paul W. Oman and Ted G. Lewis. *Milestones in Software Evolution*. Los Alamitos, CA: IEEE Computer Society Press, 1990.

Audience: Practitioner, Educator

Abstract: See Shaw87b, page 9.

[Shaw90b]

Shaw, M. Edited transcript of presentation "Workshop on Complex Software Systems," 50-52. *Strategic Directions in Computing Research; Report Based on a Conference October 11-13, 1989*. New York: Association for Computing Machinery and the Computing Research Association/ ACM Press, 1990.

Audience: Manager, Practitioner

Abstract: This report summarizes a workshop held last February under the sponsorship of the Computer Science and Technology Board of the National Research Council. The Computer Science and Technology Board set out to examine the problems of producing large complex software systems. The problems most frequently cited are those of cost overruns, schedule overruns, and software that fails to perform either to its requirement or to its intended function. But as we organized the workshop, we broadened our definition of the problem to include the opportunity costs of software that could not be created because software production capabilities were not up to the task of its creation.

[Stevens92a]

Stevens, S.M. "Next Generation Network and Operating System Requirements for Continuous Time Media." *Network and Operating System Support for Digital Audio and Video*, Ralph Herrtwich, Editor. Springer-Verlag: New York, 1992.

Audience: Practitioner

Abstract: Accessing massive multimedia databases will require multiple representations of those databases. Initial access may be through visual representations of the database. However, traversing numerous levels of tree-like structures will quickly find the user lost. Simple database queries may overwhelm users with information.

To overcome these problems, the Advanced Learning Technologies Project at Carnegie Mellon University's Software Engineering Institute embeds in multimedia objects the knowledge of the content of those objects over several dimensions. With this model, variable granularity knowledge about the domain, content, image structure, and the appropriate use of content and image is embedded with the object. In ALT, a rule base acts as a visual director, making a judgment on what image to display and how to manipulate it. This provides the ability to present disparate text, audio, images, and video intelligently in response to users' needs. This chapter discusses these techniques and provides examples of objective and subjective knowledge types for multimedia objects.

4 Work in Proceedings

- [Brown et al 92c] Brown, A.W.; Feiler, P.H.; Wallnau, K.C. "Understanding Integration in a Software Development Environment." *Proceedings of the 2nd International Conference on Systems Integration*. Morristown, NJ: IEEE Computer Society, June 1992.
Audience: Practitioner, Educator
- [Brown et al 92e] Brown, A.W.; Feiler, P.H.; Wallnau, K.C. "Past and Future Models of CASE Integration." *Proceedings of the 5th International Workshop on Computer-Aided Software Engineering (CASE'92)*. Montreal, Canada: IEEE Computer Society, July 1992.
Audience: Practitioner, Educator
- [Christel90] Christel, M.G. "A Digital Video Interactive Course on Code Inspections." *Proceedings of the Eighth Annual Conference on Technology and Innovations Training and Education*. Colorado Springs, CO: American Defense Preparedness Association, March 1990.
Audience: Practitioner, Educator
Abstract: The Software Engineering Institute is creating one of the first training applications using Intel's Digital Video Interactive (DVI) technology. This application is a course on code inspections, a formal technical review process noted for its importance to software quality by organizations such as IBM and AT&T Bell Laboratories. In addition to incorporating many new design techniques made possible by digital video, the course also includes a hypertext tool, menu-based natural language interface, and an expert system. The design and development of this course are discussed, with primary focus on those aspects of the course that are uniquely suited to the DVI medium. New interactive digital video technologies, such as DVI, CD-I, and hybrid analog and digital videodiscs, are just beginning to become commercially available. This paper communicates some lessons learned in the development of one course that future interactive digital video developers may find useful.
- [Christel92b] Christel, M.G. "Experiences with an Interactive Video Code Inspection Laboratory." Edited by Carol Sledge. To appear in the *Proceedings of the Sixth SEI Conference on Software Engineering Education*. San Diego, CA: Springer-Verlag, October 5-7, 1992.
Audience: Educator
Abstract: Software engineers need practical training in addition to classroom lectures to obtain the knowledge and skills necessary to succeed in industry. This training is provided by laboratories in other engineering disciplines. Such laboratories have been implemented as computer-based interactive video courses in the past, with numerous advantages. Based on this success, an interactive video course was created for use as a "code inspection laboratory," in which the skills of preparing for and participating in code inspections are learned and

practiced. This paper summarizes the anecdotal feedback and usage data from 120 students who used the course over the past two years. Lessons learned from these experiences are discussed, with implications for the development of future interactive video software engineering laboratories.

[Cohen 90b]

Cohen, S.G. "Locating Resources For Reuse-Based Development." *Reuse in Practice Workshop Summary. Reuse in Practice Workshop Summary, IDA Document D-754. Edited by James Baldo Jr. Alexandria, VA: Institute for Defense Analyses, July 11-13, 1989.*

Audience: Practitioner

Abstract: Locating resources in a collection of reusable software requires integration of management and retrieval methods. The maintainer of the collection will need different operations and modes of access from an application developer using the collection. Users of the collection will want access to it at different phases of the life cycle. Finally, users will have expertise in different application domains, but may desire access to the same information.

This paper briefly examines methods for managing and locating reusable resources. These methods include both attribute- and facet-based information retrieval. The paper also introduces a new method to support search and retrieval that captures information specific to an end-user application (both requirements and design) and matches that information to resources in the library. Integration of these approaches is essential for successful reuse-based software development.

[Cohen 90c]

Cohen, S.G. "Process and Products for Software Reuse in Ada." *TRI-Ada '90 Proceedings*. Baltimore, MD: ACM Publications, December 3-7, 1990.

Audience: Practitioner, Manager

Abstract: The adoption of reuse into software development will include the definition of new products and processes. On the product side, we must identify the form of deliverables that can support reuse; on the process side, we must identify the approach needed to develop and apply those products. This paper examines the processes and products that can lead to successful reuse. The first process, one widely used today, is to adapt an existing system to meet a new set of requirements. The second, a relatively new practice, identifies families of programs, providing support for parameterization of commonality and customization for unique requirements. The third process is a reuse-based engineering approach to discover and exploit commonality in software systems as the basis for software development.

[Dart89a]

Dart, S.A. "Software Development Environments from a User's Perspective." *Fifth Annual National Joint Conference on Quality and Productivity*. Alexandria, Virginia: National Joint Conference, February 1989.

Audience: Practitioner, Educator

- [Dart89b] Dart, S.A. "Tool Configuration and Installation Assistant." *Proceedings of the USENIX Workshop on Software Management*. New Orleans, LA: USENIX Association, April 1989.
Audience: Practitioner, Educator
- [Dart89c] Dart, S.A.; Feiler, P.H. "Configuration Management in CASE Tools and Environments." *Third International Workshop on Computer-Aided Software Engineering (CASE 89)*. London, England: IEEE Computer Society, July 1989.
Audience: Manager, Practitioner, Educator
- [Dart89d] Dart, S.A. "Tool Configuration Assistant." *Second International Software Configuration Management Workshop*. Princeton, NJ: ACM Press, October 24-27, 1989.
Audience: Practitioner
- [Dart91b] Dart, S.A.; Feiler, P.H. "State of the Art in Environment Support for Configuration Management." *Tutorial Notes at the 13th International Conference on Software Engineering (ICSE)*. Austin, Texas: IEEE Computer Society Press, May 1991.
Audience: Manager, Practitioner, Educator
- [Dart91c] Dart, S.A. "Concepts in Configuration Management Systems." *Third International Software Configuration Management Workshop*. Trondheim, Norway: ACM Press, June 12-14, 1991.
Audience: Manager, Practitioner, Educator
- [Dart91d] Dart, S.A. "Configuration Management in Computer Aided Design Systems." *Third International Software Configuration Management Workshop*. Trondheim, Norway: ACM Press, June 12-14, 1991.
Audience: Manager, Practitioner, Educator
- [Dart92a] Dart, S.A. "Parallels in Computer-Aided Design Framework and Software Development Environment Efforts." *Proceedings of the Third International IFIP Workshop on Electronic Design Automation Frameworks*. Paderborn, Germany: North-Holland, March 1992.
Audience: Manager, Practitioner, Educator
- [Dart92b] Dart, S.A. "The Past, Present and Future of Configuration Management." To be published in the *Proceedings of the IFIP World Congress*. Madrid, Spain: North-Holland, September 1992.
Audience: Practitioner, Educator

[D'Ippolito89a]

D'Ippolito, R.S.; Plinta, C.P. "Software Development Using Models." *Fifth International Workshop on Software Specification and Design*. Pittsburgh, PA:IEEE Computer Society Press, May 1989

Audience: Manager, Practitioner

Abstract: We have been observing the design methods of software practitioners and engineers in other disciplines and have noticed differences. The most notable difference is the lack of models available to the software practitioner. We have reached several conclusions as a result of our observations and work:

- What is currently being called software engineering is not really engineering in the traditional sense
- There are important lessons about the design process to be learned from traditional engineering disciplines
- The application of one of the key concepts in engineering, the process of *modeling*, has the potential for high payoff.

This report describes our experience and successes with modeling.

[D'Ippolito89b]

D'Ippolito, R.S. "Using Models in Software Engineering." *TRI-Ada '89*. Pittsburgh, PA: Association for Computer Machinery, October 1989.

Audience: Practitioner

Abstract: The SEI has participated in several projects in which the focus was on helping contractors make use of good software engineering methods and Ada. During this participation, we have learned several important lessons about the development of software for both large-scale and embedded systems. We have noticed that after a long period of time where the focus on productivity generated searches for new methodologies, tools, and ways to write reusable software, the emphasis has shifted to quality, in recognition of the fact that the new methods and tools were not adequate to address the problems occurring at the design level. We propose that the industry instead concentrate the search for the *old* methods still in use in the other branches of engineering, and apply those methods to the software problem.

[D'Ippolito90]

D'Ippolito, R.S. "The Context of Model-Based Software Engineering." *Proceedings of the DARPA Workshop on Domain-Specific Software Architectures*. Hidden Valley, PA: DARPA/ISTO, July 1990.

Audience: Practitioner

Abstract: The result of engineering practice is an engineered product. The behavior of an engineered product and the resources necessary to produce and operate it are predictable from the design. The prediction is possible because both the architectural elements assembled in the design and the practice elements assembled in the build plan are technology packaged as models.

[D'Ippolito92a]

D'Ippolito, R.S.; Lee, K.J. "Modeling Software Systems by Domain." *Proceedings of the AAAI-92 Workshop on Automating Software Design*. San Jose, CA: AAAI, July 1992.

Audience: Practitioner, Educator

Abstract: The software architectures engineering effort at the SEI has developed engineering modeling techniques that both reduce the complexity of software for domain-specific computer systems and result in systems that are easier to build and maintain. These techniques allow maximum freedom for system developers to apply their domain expertise to software.

We have applied these techniques to several types of applications, including training simulators operating in real time, engineering simulators operating in non-real time, and real-time embedded computer systems. Our modeling techniques result in software that mirrors both the complexity of the application and the domain knowledge requirements. We submit that the proper measure of software complexity reflects neither the number of software component units nor the code count, but the locus of and amount of domain knowledge. As a result of using these techniques, domain knowledge is isolated by fields of engineering expertise and removed from the concern of the software engineer. In this paper, we describe kinds of domain expertise, describe engineering by domains, and provide relevant examples of software developed for simulator applications using the techniques.

[D'Ippolito92b]

D'Ippolito, R.S.; Lee, K.J. "The Context of Engineering Software." To be published in *Proceedings of the 6th Conference on Software Engineering Education*. San Diego, CA: Springer-Verlag, October 1992.

Audience: Practitioner, Educator

Abstract: This paper describes a model for the engineering of products and some implications for the model. The concepts are from typical engineering disciplines. Supporting material can be found in many standard engineering texts.

[Feiler88]

Feiler, P.H.; Smeaton, R. "Managing Development of Very Large Systems: Implications on Integrated Environments." *Proceedings of the International Workshop on Software Version and Configuration Control*. Grassau, West Germany: Teubner Verlag, January 1988.

Audience: Practitioner, Educator

[Feiler90a]

Feiler, P.H. "Software Configuration Management: Advances in Software Development Environments." Tutorial. *12th International Conference on Software Engineering (ICSE 12)*. Nice, France: IEEE Computer Society Press, March 1990.

Audience: Practitioner, Educator

- [Feiler90b] Feiler, P.H. "Software Process Support in Software Development Environments." *Sixth International Software Process Workshop*. Pittsburgh, PA: IEEE Computer Society Press, October 1990.
Audience: Manager, Practitioner, Educator
- [Garlan92] Garlan, D.; Shaw, M.; Okasaki, C.; Scott, C.M.; Swonger, R.F. "Experience with a Course on Architectures for Software Systems." To appear in the *Proceedings of the Sixth SEI Conference on Software Engineering Education*. San Diego, CA: Springer-Verlag, October 5-7, 1992.
Audience: Educator
Abstract: As software systems grow in size and complexity their design problem extends beyond algorithms and data structures to issues of system design. This area receives little or no treatment in existing computer science curricula. Although courses about specific systems are usually available, there is no systematic treatment of the organizations used to assemble components into systems. These issues—the *software architecture* level of software design—are the subject of a new course that we taught for the first time in Spring 1992.

This paper describes the motivation for the course, the content and structure of the current version, and our plans for improving the next version.
- [Hardy90] Hardy, E.J.; Klein, D.V. "The Serpent UIMS." *Proceedings of the European Unix User's Group*. Nice, France: EUUG, October 1990.
Audience: Practitioner
- [Kaiser87a] Kaiser, G.E.; Feiler, P.H. "Intelligent Assistance Without Artificial Intelligence." *Proceedings of the Thirty-Second IEEE Computer Society International Conference (COMPCON)*. San Francisco, CA: IEEE Computer Society Press, February 1987.
Audience: Practitioner
- [Kaiser87b] Kaiser, G.E.; Feiler, P.H. "An Architecture for Intelligent Assistance in Software Development." *Proceedings of 9th International Conference on Software Engineering*. Monterey, CA: IEEE Computer Society Press, March 1987.
Audience: Practitioner
- [Klein87] Klein, D.V. "UBOAT – A Unix Based On-Line Aid to Tutorials." *Proceedings of the European Unix User's Group*. Dublin, Ireland: EUUG, September 1987.
Audience: Practitioner
- [Klein89a] Klein, D.V. "A Comparison of Compiler Utilization of Instruction Set Architectures." *Proceedings of the Winter 1989 Usenix Conference*. San Diego, CA: USENIX Association, January 1989.
Audience: Practitioner

- [Klein89c] Klein, D.V. "RISC vs. CISC: From the Perspective of Compiler/Instruction Set Interaction." *Proceedings of the European Unix User's Group*. Vienna, Austria: EUUG, September 1989.
Audience: Practitioner
- [Klein90a] Klein, D.V. "Foiling the Cracker: A Survey of, and Improvements to, Unix Password Security." *Proceedings of the United Kingdom Unix User's Group*. London, England: UKUUG, July 1990.
Audience: Practitioner
- [Klein90b] Klein, D.V. "Foiling the Cracker: A Survey of, and Improvements to, Unix Password Security." *Proceedings of the USENIX Security Workshop*. Portland, Oregon: USENIX Association, Summer 1990.
Audience: Practitioner
- [Klein91] Klein, D.V. "Foiling the Cracker: A Survey of, and Improvements to, Unix Password Security." *Proceedings of the 14th DoE Computer Security Group*. Concord, CA: DoE, May 1991.
Audience: Practitioner
- [Lee88a] Lee, K.J.; Plinta, C.P.; Rissman, M.S. "On Specifications and Paradigms." *AdaJUG*. Phoenix, AZ: AdaJUG, March 1988.
Audience: Practitioner
- [Lee88b] Lee, K.J., "Domain-Specific Software Architectures for C³I Systems." *AFCEA Europe Symposium*. Brussels, Belgium: AFCEA, October 1988.
Audience: Practitioner
- [Lee89a] Lee, K.J.; Rissman, M.S. "Application of Domain-Specific Software Architectures to Flight Simulator Systems." *Summer Computer Simulator Conference*. Austin, TX: Society for Computer Simulation International, July 1989.
Audience: Practitioner
- [Lee89b] Lee, K.J.; Rissman, M.S. "Application of Domain-Specific Software Architectures to Flight Simulator Systems and Training Devices." *Reuse in Practice Workshop Summary, IDA Document D-754*. Edited by James Baldo Jr. Alexandria, VA: Institute for Defense Analyses, July 11-13, 1989.
Audience: Practitioner

- [Lee90] Lee, K.J. "Engineering and Software Architectures." *Proceedings of the DARPA Workshop on Domain-Specific Software Architectures*. Hidden Valley, PA: DARPA/ISTO, July 1990.
- Audience: Practitioner**
- Abstract:** In general, the software systems planned for the next five to ten years are expected to be far larger, or far more complex, than anything successfully built to date. When we review the success rate for current large-scale systems, we conclude that the traditional ways of building such systems are unlikely to deliver well-engineered systems of the projected scale.
- Our experience helping to build systems in several important areas (flight simulators and training devices, command and control systems, MIS systems, embedded missile seeker systems) has convinced us that unless software development becomes more oriented to traditional engineering practice, most of the major problems related to software development will remain. Software engineering has much to gain by emulating the practices of systems engineering where there has been a way of designing the systems while maintaining a clear separation of domain expertise needed to produce each system component.
- [Lee92] Lee, K.J.; Cohen, S.G.; Plinta, C.P. "An Engineering Basis for Software." To be published in *Proceedings of TRI-Ada'92*. Orlando, FL: Association for Computing Machinery Press, November 1992.
- Audience: Practitioner**
- [Norman90] Norman, R.J.; Van Ghent, R.; Zarrella, P.F. "CASE Tool Integration and Standardization." *Advance Working Papers of the Fourth International Workshop on Computer-Aided Software Engineering*. Irvine, CA: IEEE Computer Society Press, December 5-8, 1990.
- Audience: Practitioner**
- [Peterson 90] Peterson, A.S. "Coming to Terms with Terminology for Software Reuse." *Reuse in Practice Workshop Summary. Reuse in Practice Workshop Summary, IDA Document D-754*. Edited by James Baldo Jr. Alexandria, VA: Institute for Defense Analyses, July 11-13, 1989.
- Audience: Practitioner**
- Abstract:** There are problems with the use of many of the terms used in software engineering when applied specifically to reuse. Three terms of particular interest, taxonomy, software reuse, and domain analysis and some problems with their usage are discussed. The specific problems with these terms are generalized and several solutions are given, the most important being the introduction of the concept of a reuse process model to provide context and an overall view of the potential areas of discourse in reuse. Several new terms are proposed for future use as well as definitions of existing terms that are meaningful in the context of software reuse.

- [Plinta89a] Plinta, C.P.; Lee, K.J. "A Model Solution for C³I Systems." *Reuse in Practice Workshop Summary, IDA Document D-754*. Edited by James Baldo Jr. Alexandria, VA: Institute for Defense Analyses, July 11-13, 1989.
- Audience: Practitioner**
- Abstract:** This paper briefly describes a specific portion of recent work performed by the domain-specific software architectures efforts at the SEI-the development and use of a model solution for message translation and validation in the C³I domain. Based on this experience and our involvement with programs in the C³I domain, future considerations are described. These consideration involve identifying potential models within a domain and making recommendations for developing and documenting model solutions that will enable the models to be reused.
- [Plint89b] Plinta, C.P.; Lee, K.J. "A Model Solution for C³I Systems." *TRI-Ada '89*. Pittsburgh, PA: Association for Computing Machinery, November 1989.
- Audience: Practitioner**
- Abstract:** See Plinta89a, page 20.
- [Plinta90] Plinta, C.P. "Considerations for Defining Software Architectures." *Proceedings of the DARPA Workshop on Domain-Specific Software Architectures*. Hidden Valley, PA: DARPA/ISTO, July 1990.
- Audience: Practitioner**
- Abstract:** This paper provides guidance for identifying candidate architectures based on the author's experience. Two approaches to finding an architecture for an application area are discussed: an archeological approach and a contextual approach.
- [Rader92] Rader, J.A. "Pro Rebuttal: The Investment in CASE Tools Is Warranted by the Productivity Gain of the Development Staff." *Proceedings Achieving Quality Software Debate*. San Diego, CA: Society for Software Quality, January 29-31, 1992.
- Audience: Manager, Practitioner**
- [Schaefer92] Schaefer, W.; Shaw, M. "Design Methods and Software Processes." *Proceedings of the Sixth International Workshop on Software Specification and Design*. Como, Italy: IEEE Computer Society Press, October 25-26, 1991.
- Audience: Practitioner, Educator**

- [Shaw85a] Shaw, M. "An Input-Output Model for Interactive Systems." *Proceedings of the NGI/SION Annual Symposium: Conference on Human Factors in Computer Systems*. Utrecht, Netherlands: Nederlands Genootschap voor Informatica, Stichting Informatica, April 1-2, 1985.
Audience: Practitioner
Abstract: Interactive user interfaces depend critically on underlying computing system facilities for input and output. However, most computing systems still have input-output facilities designed for batch processing. These facilities are not adequate for interfaces that rely on graphical output, interactive input, or software constructed with modern methodologies. This paper details the deficiencies of batch-style input-output for modern interactive systems, presents a new model for input-output that overcomes these deficiencies, and suggests software organizations to take advantage of the new model.
- [Shaw85b] Shaw, M. "What Can We Specify? Issues in the Domains of Software Specifications." *Proceedings of the Third International Workshop on Software Specification and Design*. London, UK: IEEE Computer Society Press, August 26-27, 1985.
Audience: Practitioner
Abstract: Formal specifications customarily deal exclusively with the domain of functional properties of software. However, other domains are of interest to software designers and developers. Two particular areas of concern for practical software development are not yet well served by formal specifications. This paper raises issues about how these areas might be better served.
- [Shaw86a] Shaw, M. "An Input-Output Model for Interactive Systems." *Proceedings of CHI'86: Conference on Human Factors in Computing Systems*. Boston, MA: ACM SIGCHI, April 13-17, 1986.
Audience: Practitioner
Abstract: See Shaw85a, page 21.
- [Shaw86c] Shaw, M. "Beyond Programming-in-the-Large: The Next Challenges for Software Engineering." *Advanced Programming Environments: Proceedings of an International Workshop*. Trondheim, Norway: Springer-Verlag, June 16-18, 1986.
Audience: Practitioner, Educator
Abstract: See Shaw86b, page 5.
- [Shaw87d] Shaw, M. "Purposes and Varieties of Software Reuse." *Proceedings of Tenth Minnowbrook Software Workshop*. Blue Mountain Lake, NY: Syracuse University, July 28-31, 1987.
Audience: Educator
Abstract: In the past year or two, software reuse has gained fresh attention as a technique for dealing with some of the problems of software development and maintenance. Unfortunately, much of the acclaim is uncritical, the subject is more complex than many of its advo-

cates realize, and thirty years of indifferently successful history is often ignored. As a result, "software reuse" is in danger of becoming the empty buzzword of the late 1980s.

This paper sharpens the focus on the problems that software reuse might solve and how it might do so, and proposes some structure for the subject area in order to bring out some of the (very different) concerns of various projects and to make objective comparisons of different techniques possible.

[Shaw88b]

Shaw, M. "Toward Higher-Level Abstractions for Software Systems." *Proceedings of Tercer Simposio Internacional del Conocimiento y su Ingenieria*. Polytechnical University of Madrid, Spain: Rank Xerox, October 17-21, 1988.

Audience: Practitioner, Educator

Abstract: Software now accounts for most of the cost of computer-based systems. Over the past thirty years, abstraction techniques such as high-level programming languages and abstract data types have improved our ability to develop software. However, the increasing size and complexity of software systems have introduced new problems that are not solved by the current techniques. These problems involve the system-level design of software in which the important decisions are concerned with the kinds of modules and subsystems to use and the way these modules and subsystems are organized. This level of organization, the software architecture level, requires new kinds of abstractions. These new abstractions will capture essential properties of major subsystems and the ways they interact.

[Shaw89c]

Shaw, M. "Remembrances of a Graduate Student." *Proceedings of the 11th International Conference on Software Engineering*. Pittsburgh, PA: IEEE Computer Society Press, May 15-18, 1989.

Audience: Educator

Abstract: See Shaw89b, page 5.

[Shaw89d]

Shaw, M. "Larger Scale Systems Require Higher-Level Abstractions." *Proceedings of the Fifth International Workshop on Software Specification and Design*. Pittsburgh, PA: IEEE Computer Society Press, May 1989.

Audience: Practitioner, Educator

Abstract: Over the past thirty years, abstraction techniques such as high-level programming languages and abstract data types have improved our ability to specify and develop software. However, the increasing size and complexity of software systems have introduced new problems that are not solved by the current techniques. These new problems involve the system-level design of software in which the important decisions are concerned with the kinds of modules and subsystems to use and the way these modules and subsystems are organized. This level of organization, the software architecture level, requires new kinds of abstractions that capture essential properties of

major subsystems and the ways they interact. This paper discusses classical system organizations, the building blocks from which they are constructed, and a programming-language approach to specification.

[Shaw90c]

Shaw, M. "An Input-Output Model for Interactive Systems." *Visual Programming Environments: Applications and Issues*. Los Alamitos, CA: IEEE Computer Society Press, MONTH, 1990.

Audience: Practitioner, Educator

Abstract: See Shaw85a, page 21.

[Shaw90e]

Shaw, M. "Elements of a Design Language for Software Architecture." *IEEE Design Automation Workshop on System Level Modelling*. Scottsdale, AZ: IEEE, January 21-24, 1990.

Audience: Practitioner, Educator

Abstract: Hardware design is carried out at several different levels, with different issues, models, and design strategies at each level. The same situation is true for software, except that the distinctions among levels—or even the existence of distinct levels—is not as widely recognized. A delivered software system should consist of a model for each of these levels, each expressed in some appropriate language.

[Shaw90g]

Shaw, M.; Perry, J.M. "The Role of Domain Independence in Promoting Software Reuse: Architectural Analysis of Systems." *Reuse in Practice Workshop Summary, IDA Document D-754*. Edited by James Baldo Jr. Alexandria, VA: Institute for Defense Analyses, July 11-'3, 1989.

Audience: Practitioner

Abstract: Domain analysis for reuse is a topic of much current interest and study. While there are several variations of domain analysis, they are usually characterized by their emphasis on application dependencies. This position paper describes architectural analysis which is a type of analysis for furthering our understanding of software architectures. It attempts to raise the abstraction level of design elements and, thereby, emphasizes domain independence. Although architectural analysis and domain analysis for reuse have different processes and goals, they are closely related and support one another. This mutual support is identified and examined.

[Shaw91b]

Shaw, M.; Tomayko, J.E. "Models for Undergraduate Project Courses in Software Engineering." *Curriculum Design Workshop*. Cambridge, MA: MIT, January 10-12, 1991.

Audience: Educator

Abstract: The software engineering course provides undergraduates with an opportunity to learn something about real-world software development. Since software engineering is far from being a mature engineering discipline, it is not possible to define a completely satisfactory syllabus. Content with a sound basis is in short supply, and the material most often taught is at high risk of becoming obsolete within a

few years.

Undergraduate software engineering courses are now offered in more than a hundred universities. Although three textbooks dominate the market, there is not yet consensus on the scope and form of the course. The two major decisions an instructor faces are the balance between technical and management topics and the relation between the lecture and project components. We discuss these two decisions, with support from sample syllabi and survey data on course offerings in the United States and Canada. We also offer some advice on the management of a project-oriented course.

- [Shaw91c] Shaw, M. "Informatics for a New Century: Computing Education for the 1990s and Beyond." *IFIP Working Group 3.2 International Workshop on Informatics Education in the 1990s*. Providence, RI: North-Holland, April 5-7, 1990.
Audience: Educator
Abstract: See Shaw91a, page 10.
- [Shaw91d] Shaw, M. "Putting Engineering into Software Engineering Education." Position paper for the *IEEE Software Engineering Software Engineering Workshop at the 13th International Conference on Software Engineering*. Austin, TX: IEEE Computer Society Press, May 13-17, 1991.
Audience: Practitioner, Educator
Abstract: The current practice of software engineering bears only slight resemblance to the usual standards of engineering practice. This is in part a consequence of the immaturity of the field, but it also results from the failure of software engineering education to instill an engineering mindset in students. This position paper discusses the character of engineering practice with special emphasis on the routine use of reference materials; based on this characterization, it suggests ways software engineering education could better support good engineering practice.
- [Shaw91e] Shaw, M.; Tomayko, J.E. "Models for Undergraduate Project Courses in Software Engineering." *Proceedings of the Fifth SEI Conference on Software Engineering Education*. Pittsburgh, PA: Springer-Verlag, October 7-8, 1991.
Audience: Educator
Abstract: See Shaw91b, page 11.
- [Shaw91f] Shaw, M. "Heterogeneous Design Idioms for Software Architecture." *Proceedings of the Sixth International Workshop on Software Specification and Design*. Como, Italy: IEEE Computer Society, October 25-26, 1991.
Audience: Practitioner
Abstract: Software designers use a variety of structural patterns to specify system architectures. These patterns, or idioms, are currently

used informally and imprecisely. Nevertheless, they provide a useful, broadly shared vocabulary. In practice, a given design often relies on several patterns. This paper reviews some common architectural idioms, shows several ways in which they are used heterogeneously, and discusses the benefits of making these idioms and their combinations more explicit and precise.

[Shaw92b]

Shaw, M.; Wulf, W. "Tyrannical Languages *Still* Preempt System Design." *Proceedings of the 1992 International Conference on Computer Languages*. San Francisco, CA: IEEE Computer Society Press, April 20-23, 1992.

Audience: Practitioner, Educator

Abstract: It is a prime tenet of most programming language design that "higher-level" languages are a good thing – indeed the higher the level, the better. The assumption is that the higher the level of the language – the more abstract the abstractions – the greater the leverage provided to the programmer. Language designers usually ensure that the higher-level constructs capture their intention by completely specifying the associated semantics.

A decade ago, we challenged the "higher-level is better" assumption. The paper in which we did this has largely been ignored. In fact we see this apparently benign assumption as aggressively interfering with good application design. Unfortunately, the consequences of blind adherence to this tenet are spreading in both current language proposals and larger system designs. This paper re-examines the claim that "higher-level" is better and the counter-claim that "higher-level" constructs interfere with software development when they preempt the developer's design prerogatives. The starting point for discussion is our earlier paper, "Toward Relaxing Assumptions in Languages and Their Implementations".

[Stevens91]

Stevens, S.M., & Christel, M.G. "Embedding Knowledge in Continuous Time Media." *Conference on System Support for Continuous Time Media*. Pittsburgh, PA: Information Technology Center/International Business Machines, 1991.

Audience: Practitioner

Abstract: It is difficult to move through information that has an intrinsic and essentially fixed temporal element such as video. While detailed indexing of video can help, users often wish to peruse video much as they flip through the pages of a book. This paper highlights two techniques developed for the Advanced Learning Technologies Project that will facilitate such searches. First, detailed, embedded knowledge of the video information will allow for scans by various views, such as by content area or depth of information. Second, partitioning multimedia data into smaller objects reduces bandwidth problems associated with accessing central data in large video files. Concatenation of logically contiguous files allows for seamless, continuous play of long sequences.

[Stevens92c]

Stevens, S.M. "Interface, Fidelity, and Development Issues in a Virtual Reality Workspace for Software Engineering." To be published in *Man/Machine Interaction, Autumn School '92*. Paris, France: Thomson-lepens, 1992.

Audience: Practitioner, Educator

Abstract: Using empirical and anecdotal data this paper discusses virtual reality interface design and testing including methods of navigation (both physical and information spaces), natural language interfaces, and image composition. Fidelity issues from high level design of virtual worlds to image resolution and frame rate are discussed.

[Stewart90]

Stewart, J.A. "An Example of the Re-Application of a Model-Based Architecture." *Proceedings of the DARPA Workshop on Domain-Specific Software Architectures*. Hidden Valley, PA: DARPA/ISTO, July 1990.

Audience: Practitioner

Abstract: The software architectures engineering effort has been developing ideas about software architectures and model-based design issues for several years. During this period, both models and architectures (assemblages of models) have been developed for several projects. This article discusses the transition of an existing model to a new application area and the architecture that resulted.

5 Acknowledgments

Many thanks to all the authors listed for their assistance and contributions toward the creation of this bibliography. In addition, special thanks go to those who reviewed this work, in particular, Dan Bidwa, Mike Christel, Tim Herren, Fred Long, Jeff Stewart, Judy Vernick, Jim With-ey, and especially to the technical writer/editor, Sandra Bond, for her expertise and guidance.

Appendix A Information about the Software Engineering Institute

For more information about the Software Engineering Institute, or to obtain a copy of the current version of "SEI Reports: Annotated Listing" (a listing of SEI reports that are available for public distribution), contact:

Kristine Peebles
SEI Customer Relations
kdp@sei.cmu.edu
(412) 268-5800

To obtain copies of SEI reports that are available for public distribution, please contact one of the agencies listed below.

- RAI:** Research Access Inc.
3400 Forbes Avenue
Suite 302
Pittsburgh, PA 15213
Telephone: 800-685-6510
or (412) 682-6510
FAX: (412) 682-6530
- NTIS:** National Technical Information Service
U.S. Department of Commerce
Springfield, VA 22161-2103
Telephone: (703) 487-4600
- DTIC:** Defense Technical Information Center
ATTN: FDRA Cameron Station
Alexandria, VA 22304-6145
Telephone: (703) 274-7633

Appendix B Members of the SEI Engineering Techniques Program as of July 1992

Joanne Beckman
Diane Bradley
Sandy Brenner
Alan Brown
Michael Caldwell*
Michael Christel
Alan Christie
Sholom Cohen
Susan Dart
Jane Walter DeSimone
Richard D'Ippolito
Peter Feiler
Tim Herren*
Clifford Huff
Kyo Kang
Daniel Klein
Robert Krut
Kenneth Lee
Cathy Lin*
Fred Long **
Edwin Morris
Lorraine Nemeth
A. Spencer Peterson
Patrick Place
Chuck Plinta
Jock Rader*
Mary Shaw
Dennis Smith
Jay Stanley*
Scott Stevens
Jeffrey Stewart
James Withey
Dave Wood
Paul Zarrella

* Resident Affiliate

** Visiting Scientist

Index by Author

B

Barbacci, M. 3
Barghouti, N.S. 4
Brookes, S.D. 8
Brown, A.W. 3, 8, 12

C

Christel, M.G. 3, 12
Cohen, S.G. 8, 13, 19

D

D'Ippolito, R.S. 15, 16
Dart, S.A. 3, 9, 13, 14
Donner, M. 8
Driscoll, J. 8
Druffel, L. 9

E

Earl, A.N. 8

F

Feiler, P.H. 4, 12, 14, 16, 17

G

Garlan, D. 17

H

Habermann, A.N. 3
Hardy, E.J. 17
Huff, C.C. 4

K

Kaiser, G.E. 4, 17
Klein, D.V. 4, 17, 18

L

Lee, K.J. 16, 18, 19, 20

M

Mauldin, M. 8
McDermid, J.A. 3, 8

N

Norman, R.J. 19

O

Okasaki, C. 17
Oman, P. 7

P

Palmer, C. 8
Pausch, R. 8
Penedo, M.H. 3
Perry, J.M. 23
Peterson, A.S. 4
Plinta, C.P. 15, 18, 19, 20

R

Rader, J.A. 20
Rissman, M.S. 18

S

Schaefer, W. 5, 20
Scherlis, W.L. 8
Schwanke, R.W. 4
Scott, C.M. 17
Shaw, M. 3, 5, 6, 7, 8, 9, 10, 11, 17, 20, 21, 22, 23, 24, 25
Smeaton, R. 16
Smith, D. 7
Spector, A. 8
Stevens, S.M. 7, 11, 25, 26
Stewart, J.A. 26
Swonger, R.F. 17

T

Tomayko, J.E. 23, 24

V

Van Ghent, R. 19

W

Wallnau, K.C. 12
Wulf, W. 25

Z

Zarrella, P.F. 19

Index by Target Audience

Manager

B
Barbacci85 3
Brown et al 92d 3

C
Cohen90a 8

D
D'Ippolito89a 15
Dart89c 14
Dart91a 9
Dart91b 14
Dart91c 14
Dart91d 14

F
Feiler90b 17

H
Huff92 4

R
Rader92 20

S
Shaw86b 5
Shaw87e 5
Shaw90f 6
Shaw90h 6, 11
Smith90 7

Practitioner

B
Brown et al 92a 3, 8
Brown et al 92b 3
Brown et al 92c 8, 12
Brown et al 92e 12

C
Christel90 12
Christel92a 3
Cohen 90b 13
Cohen 90c 13

D
D'Ippolito89a 15
D'Ippolito89b 15
D'Ippolito90 15
D'Ippolito92a 16
D'Ippolito92b 16
Dart89a 13, 14
Dart89b 14
Dart89d 14
Dart89e 3
Dart91a 9
Dart91b 14
Dart91c 14
Dart91d 14
Dart92a 14
Dart92b 14

F
Feiler88 16
Feiler90a 16
Feiler90b 17

H
Hardy90 17
Huff92 4

K
Kaiser87a 17
Kaiser87b 17
Kaiser88a 4
Kaiser88b 4
Klein87 17
Klein89a 17
Klein89b 4
Klein89c 18
Klein90a 18
Klein90b 18
Klein91 18

L
Lee88a 18
Lee88b 18
Lee89a 18
Lee89b 18
Lee90 19
Lee92 19

N	
Norman90	19
P	
Peterson 90	19
Peterson91	4
Plinta89a	20
Plinta89b	20
Plinta90	20
R	
Rader92	20
S	
Schaefer92	5, 20
Shaw85a	21
Shaw85b	21
Shaw86a	21
Shaw86b	5
Shaw86c	21
Shaw87b	9
Shaw87c	10
Shaw87e	5
Shaw88a	5
Shaw88b	22
Shaw89a	10
Shaw89d	5, 22
Shaw90a	10
Shaw90b	11
Shaw90c	23
Shaw90d	6
Shaw90e	23
Shaw90f	6
Shaw90g	23
Shaw90h	6
Shaw91d	24
Shaw91f	24
Shaw92b	25
Smith90	7
Stevens89	7
Stevens91	25, 26
Stevens92a	11
Stevens92b	7
Stevens92c	26
Stewart90	26

Educator	
B	
Brookes et al 85	8
Brown et al 92a	8
Brown et al 92c	12
Brown et al 92e	12
C	
Christel90	12
Christel92a	3
Christel92b	12
Cohen90a	8
D	
D'Ippolito92a	16
D'Ippolito92b	16
Dart89a	13
Dart89b	14
Dart89c	14
Dart91a	9
Dart91b	14
Dart91c	14
Dart91d	14
Dart92a	14
Dart92b	14
F	
Feiler88	16
Feiler90a	16
Feiler90b	17
G	
Garlan92	17
K	
Kaiser88a	4
P	
Peterson91	4
S	
Schaefer92	5, 20
Shaw86b	5
Shaw86c	21
Shaw87a	9
Shaw87b	9
Shaw87c	10
Shaw87d	21

Shaw87e	5
Shaw88a	5
Shaw88b	22
Shaw89a	10
Shaw89b	5
Shaw89c	22
Shaw89d	22
Shaw89e	5
Shaw90a	10
Shaw90c	23
Shaw90d	6
Shaw90e	23
Shaw90f	6
Shaw90h	6
Shaw91a	6
Shaw91b	23
Shaw91c	24
Shaw91d	24
Shaw91e	24
Shaw92a	7
Shaw92b	5, 25
Stevens89	7
Stevens92a	11
Stevens92c	26

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SEI/CMU-92-SR-10			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute		6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office		
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			7b. ADDRESS (City, State and ZIP Code) ESC/AVS Hanscom Air Force Base, MA 01731		
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office		8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003		
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO N/A
			WORK UNIT NO. N/A		
11. TITLE (Include Security Classification) A Bibliography of Externally Published Works by the SEI Engineering Techniques Program					
12. PERSONAL AUTHOR(S) Sandy Brenner and Gibbie Hart					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM TO		14. DATE OF REPORT (Yr, Mo., Day) August 1992	
15. PAGE COUNT 38 pp.					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number) software engineering bibliography bibliography		
FIELD	GROUP	SUB. GR.			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This bibliography lists works by the members of the Software Engineering Institute (SEI) Engineering Techniques Program that are not published or available from the SEI. The bibliography is organized by type of work (i.e., journal or magazine article, book, or proceedings.) Indices are provided by author and targeted audience.					
(please turn over)					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPTDTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution		
22a. NAME OF RESPONSIBLE INDIVIDUAL John S. Herman, Capt, USAF			22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631		22c. OFFICE SYMBOL ESC/AVS (SEI)

ABSTRACT —continued from page one, block 19